Letterset Manipulation Utilities

REVISED JANUARY 1ST, 1983

RCM Computers

881 HIRSCHFIELD DR.
WILLIAMSVILLE, NY 14221
(716) 634 - 3088

LETTERSET MANIPULATION UTILITIES

Contents

LETTERSET MANIPULATION UTILITY INSTRUCTIONS

The following documentation is designed to describe the use of our new graphic Manipulation routines. These are the same routines we have been using in-house for some time. Their use requires that the user be familiar, if not proficient, with the use of our GEAP and Dot Writer packages. This documentation is complete, however, it is assumed that the user has a good understanding of both our programs and his own Computer and DOS. We will document only our programs and not the various Dos' that you might care to run them under.

The programs are BASIC control modules with MACHINE LANGUAGE overlays. They load like BASIC programs but each routine will call its own support programs. For this reason, the entire group of programs must be on line when in use. It is permissible to begin a program and then to remove the diskette and replace it with another disk when the object file is called for. In most cases, this will work, however, we recommend that two drives be used for the most efficient operation of the utility.

Your utility disk contains the following programs:

1.	SUSTRNG	12.	RPL	23.	MN
2.	MGE	13.	MINIMUM	24.	MAGEX
3.	CL	14.	WIGGLE	25.	MG
4.	IT	15.	ITALIC	26.	BL
5.	BLANKET	16.	WGLE/ML	27.	ROTATE
6.	UPDOWNML/CIM	17.	MAGEX/ML	28.	FILTLET
7.	UPDOWN	18.	COPYLET	29.	DEBUGLET
8.	SPLIT	19.	ITALML/CIM	30.	UD
9.	LETMENU	20.	FL	31.	SPML/CIM
10.	DB	21.	CLLTML/CIM	32.	RT
11.	SP	22.	FORMATLF		

RUNNING THE MANIPULATION PROGRAM

Each letterset / graphic manipulation routine is a separate program. That should be obvious by the filename list above. For ease of operation these programs have been linked with a program called "LETMENU". LETMENU is a menu of the facilities, listed by function, not program name. Each routine or series of routines can be selected from LETMENU and, in turn, each routine ends with the option to RERUN, RETURN TO MENU, or END. RERUN restarts the utility just used, RETURN TO MENU will take you back to LETMENU and END brings up a BASIC READY prompt.

Running the "LETMENU" program is quite simple. Bring up BASIC, be sure that you specified four (4) files, and RUN "LETMENU". There is no need to set memory size. Each DOS has its own method for running BASIC and setting files so check your DOS for this information. Here is an example of the setup using DOSPLUS 3. 4. :

From DOS READY type: BASIC LETMENU-F: 4 <ENTER>.

This command format will load BASIC with 4 files set. It will also load and run the "LETMENU" program. AGAIN - THIS FORMAT APPLIES ONLY TO DOSPLUS. You must follow the procedure required by your own DOS!

THE LETMENU DISPLAY

Following is a display of the same menu you will see on your monitor when you run the "LETMENU" program:

Letter Manipulation Utilities

1 split letter into pieces, magnify pieces
2 copy between lettersets
3 format new lettersets
4 list blanks in letterset to printer
5 list byte dump of letter(s) to printer
6 italicize letterset
7 move letter(s) within frame, and/or justify or center
8 merge string with letter(s)
9 expand or magnify letter(s) within frame
10 add wiggle to letter(s)
11 rotate and/or mirror image letter
12 find minimum storage values
13 run DOT WRITER
<ENTER> number of your choice . . .

In order to select an option, simply type the number of the routine and hit <ENTER>. The desired routine will automatically load and will in turn, load its own routines.

Now let's take a look at each routine and briefly examine its operation. Remember, we will expect that you are already familiar with the DOT WRITE / GEA graphic creation routines. We will not spend any time explaining those facilities.

CHOICE 1, THE SPLIT ROUTINE

The split routine will allow you to enlarge and refine a graphic in the following fashion. First, remember that each graphic is designed in a frame. For point of example we will use the full screen as the frame but, the actual frame can be any size from 1 dot wide by eight dots high, to 128 dots wide by 48 dots high. Also remember that, even though the width can be incremented by single dot units, the height must be incremented by printer lines where 1 line equals 8 dots high.

Once you have designed your graphic, be it a letter or a picture, you will assign it to a keyboard character. For our purposes we will always assign the original graphic to the "! " character. With this character saved under a filename (ALPHA for example) you can now run the SPLIT program.

When run, SPLIT will ask you for the file containing the graphic you wish to split. In this case, reply ALPHA. Next, you will be asked for the keyboard character that the graphic was saved under. Reply "! " (don't include the quotes). Finally, you will be asked for a filename for the output file. We will use BETA.

Now the SPLIT program begins its work. You will see what is happening on the video display. As an aside, when we first started working with these programs they were all VERY slow. Some were so slow that you could start the routine and go out to dinner. When you came back, providing you're a slow eater, the routine would be finished. You will see however, that the routines are all very fast now and many occur so quickly they will be hard to follow.

The SPLIT program will first divide the graphic FRAME into four parts. Each of the four parts will then be magnified PROPORTIONALLY to fill the entire size of the original frame. The resulting four quarters, which are now each as big as the original, are stored under the keyboard characters ", #, $, %. No file creation is done at this point.

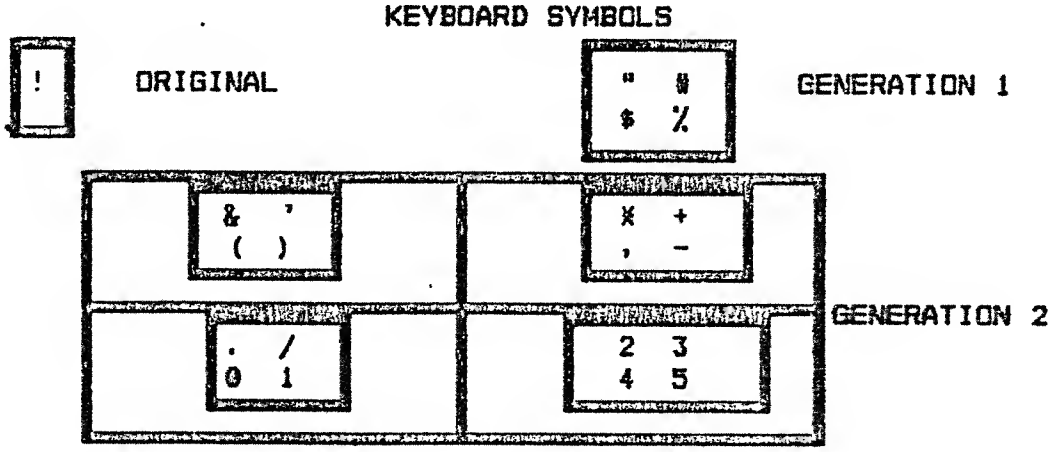
SPLIT will now ask you if you want the procedure to be run again. If you respond NO, the new graphics will be stored under the above keyboard characters and placed in the file we named BETA. Each part will be as large as the original, and each can be loaded separately for editing. In order to print out the larger graphic, you would use either GEAP or DOTPRINT and would load file BETA with either the H command (GEAP) or the . BF BETA command (DOTPRINT). Then, by printing the four sections in order:

"#

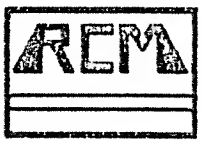
$%

you will be able to recreate the original graphic only 4 times as large.

If you answer YES to the RUN AGAIN prompt, the entire procedure will be rerun on each of the four quarters, making a 16 times larger diagram. Following is a layout of the characters that each segment will be saved under, in their relative positions:

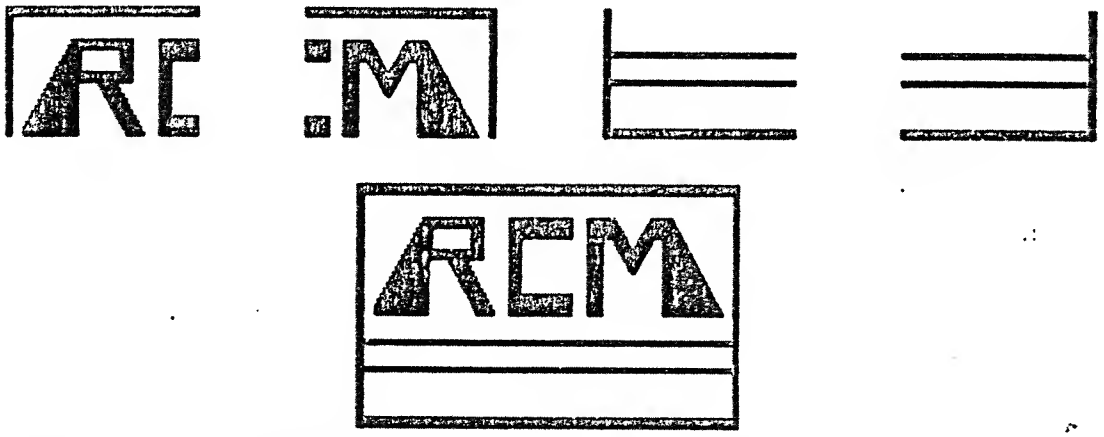


Now let's look at an example of how the SPLIT program works. First we start with our drawing in file ALPHA stored under character "! ". I'll use the . BF command to load ALPHA.

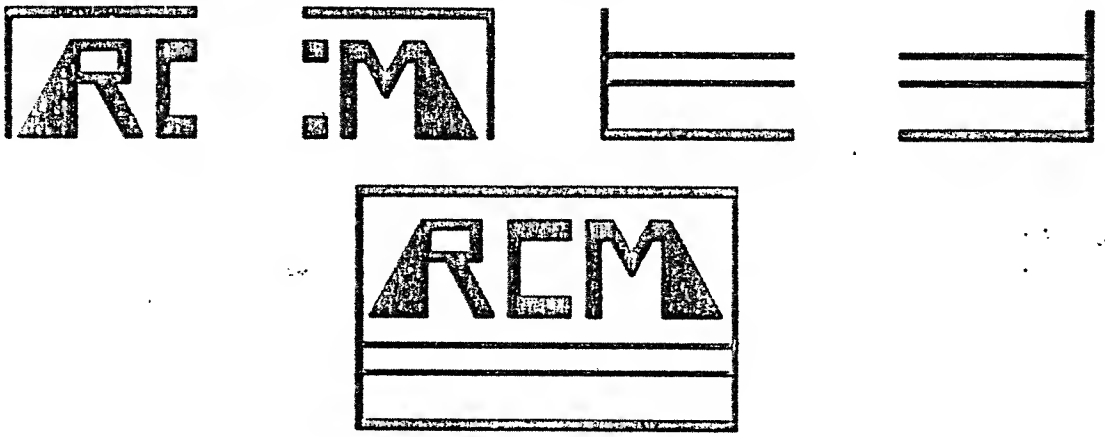


This little logo was created on one screen of GEAP. Now I'll SPLIT it into 4 parts and display each of the 4 parts separately and then together. This will be file BETA, and the characters will be " # $ and %. First I'll display them in that order, then in the proper order:

"#
$%



Note that each of the four parts is now the same size as the entire original graphic. You will also note that the first SPLIT of the graphic caused a stair step effect in the angled sides of the graphic. Now, with a little editing with GEAP and Dot Writer, I can smooth out the rough edges.



Every magnification will increase the BLOCK appearance of the resulting raphic. Be sure to edit the result of the magnification to smooth out the lines nd round out the edges.

The 16 time split works in exactly the same way. Refer to the diagram for he character layout. Also, when you use the 16 times split, the character edges ill be even rougher than with the 4 times split. Again, each resulting character an be edited and this is an easy procedure since the split sections are roportional. All edges will continue to match up.

That is all there is to it. With the SPLIT routine you can easily create a raphic up to 16 screens in size. Now let's move on to the next routine.

In the discussion of Option #1, the SPLIT routine, I gave an example for you o follow. In most of the following sections I will not give you an example but will xplain the use and prompting procedure. Later in the manual, you will see how hese routines can be used together to create lettersets and advanced graphics

While some routines like SPLIT and WIGGLE can be best explained by example, thers, such as COPY LETTERS and FORMAT are too simple to demonstrate. For e, the FORMAT routine formats a file to your specifications. You tell it how nany dots wide and how many printer lines high the file should be and it does the est. There is really nothing to demonstrate.

Don't worry though. I'm not going to leave you out in the cold. For the routines that fall into the SIMPLE category, I have made sure that there are adequate examples of their use in conjunction with the other routines.

CHOICE 2, THE COPY LETTER ROUTINE

This is one of the simple routines I spoke of. No glamour here! With COPY you can copy the letters (any one or group of letters) from one file to another. This makes it possible to take the lowercase characters from one font and place them nto the position of the lowercase characters of another font. The only requirement is that the receiving file frame size must be large enough to hold the ncoming letters. Don't get me wrong! You can copy a large letter into a small rame - but the result will be a chopped off letter.

LETTERSET MANIPULATION UTILITIES

Let's take a look at the procedure. First, get an ASCII character order chart. There is one in the GEAP manual, one in the Dot Writer manual, it is also printed on the "cheat sheet" that comes with the Dot Writer manual and, finally, I have included one of the Dot Writer "cheat sheets" with this manual so you will be certain to have the info! Remember that keyboard characters, regardless of what you have stored under them, are stored on the disk in ASCII order from the LOWEST value to the HIGHEST value. That means (look at the chart) that the "! " is the first character in the file and the "z" is the last!

OK! Now that that's out of the way, we'll select option #2 from the menu. You are prompted for the name of the file to be copied from: That is the INPUT FILE. Type in the name of your SOURCE file (that's where you want the letters to come from). Next you will be asked the name of the file that the copied letters are to be sent to: That is the OUTPUT FILE. If the OUTPUT file is the same as the INPUT file (for example, if you want to duplicate the uppercase characters into the lowercase positions) you need only enter -1. That will make both the INPUT and OUTPUT file the same.

Now that you have the files specified, it is time to specify the characters to be copied. In order to specify the range of characters, you must know what order characters are stored in. That means you must use your ASCII chart. For argument's sake we'll assume that we are going to transfer the lowercase of file B into the lowercase position of file A. The INPUT file is "B" - that is the source. The OUTPUT file is "A" - that is the destination.

Now you will be asked for the LOWEST character to be copied. That means: What is the LOWEST (in ASCII value) character you want to take from the source (B) file? Since we are copying the lowercase, the answer would be??? Right! the answer is "a". Look at your chart and see that, of all the lowercase characters, "a" has the lowest ASCII value (97). You will further note that the highest ASCII value is "z" with a whopping 122. Isn't it convenient how they came out in order? So, for the LOWEST character to be copied you will respond "a".

Your next prompt will be for the HIGHEST character to be copied. Again, this refers to highest in ASCII value and of course, the answer in our example will be "z". Now we have specified the range. In fact, our range is the entire lowercase "a" to "z". If we had wanted to copy only one character we would simply specify the same character for both LOWEST and HIGHEST. Easy isn't it?

Now for the last step. All we have to do now is to tell the COPY routine where we want our range of characters copied to. We have given it the filename so we need only tell it where in that file to store the letters. We have also told the COPY program the range of letters (how many) we want to copy. All that is left is to tell the COPY program where to place the first (LOWEST) character. The final prompt is for the LOWEST character to be OUTPUT. The program is asking you for the location where it should start placing the range of letters we selected. Since we are copying the lowercase into the lowercase position, it would be a good idea to place the "a" into the "a" position. So the correct answer to this final prompt is "a". Now the program will do the rest. The "b" will be stored under "b", "c" under "c", and so on until the range is complete and the copy function is done.

Just suppose for a moment that we had answered "A" for the final prompt instead of "a". What would have happened? No, don't guess, because I'm going to tell you anyway. If we had made that error, the entire lowercase of file "B" would have been stored in the uppercase locations of file "A". When we typed out the file later on, the resulting text would be somewhat disrupted.

Not much to it is there? Well, don't be deceived! Once you begin using this package you will find this little COPY routine very valuable. Suppose you create a font of all fancy, uppercase letters. You have spent hours detailing and debugging this work of art. Now you sit down to type and you realize that you must ALWAYS remember that when you use this font, the entire text MUST be in uppercase. If you let a lowercase letter slip in you will not get a beautiful uppercase replacement. You will get garbage. Well, copy the uppercase into the lowercase positions dummy! There, one less thing to worry about.

Now let me briefly tell you of the foibles of the COPY program. First, if you copy a letter from a small frame size file into a large frame size file you will get a complete and accurate transfer. However, the small letter will be padded to fit the larger frame and will not necessarily look good with its new frame. The closer in frame size two lettersets are, the better they will look together. Second, if you transfer in the reverse direction, a large frame size into a smaller frame size, the larger letter may be cut off. I say "MAY BE CUT OFF" because a letterset does not absolutely have to be in an exact fit frame. If the larger letter is only a little larger, it may be OK. However, if you copy Olde English into the Microprint frame, I can guarantee you that the result will be garbage!

There is a problem with using this routine. It only works on a clean disk. If garbage has been saved into a letterset file, or if an aborted letter creation has been saved, that garbage or part letter will be counted as a letter. The best way to avoid this problem is, when you create a letter that you don't like and you want to try it again at a later date, save a BLANK frame to that key location on disk. The method for doing this is part of the DOT WRITER program.

CHOICE 5, THE BYTE DUMP ROUTINE

Again, this is a simple routine. The object is to dump a list of the actual byte layout of a graphic or range of letters. On the EPSON printer, the letter itself is actually printed. The operation is self prompting. You will be asked for the filename of the object file. Then you will be asked for the LOWEST letter and then the HIGHEST letter to be dumped. The range you specify is based on the ASCII table you have been supplied. If you wanted to dump a whole letterset, you would specify "! " as the LOWEST letter and "z" as the highest. If you only wanted to dump one letter, say the "B", you would specify that letter for both LOWEST and HIGHEST letters.

This particular utility is designed for the advanced user and has little practical application for the average user. One of our main uses is when we copyright a letterset. We copyright the BYTE DUMP as well as the appearance of the letter and the name. This gives us double protection.

The remaining utilities (with the exception of number 12) are relatively difficult to use. Rather than crash on through, I'll start a new section for the remainder of the utilities.

CHOICE 6, THE ITALICIZE ROUTINE

The italics option is relatively easy to use, however, it offers some unique challenges to the user. Let's first consider the prompt sequence.

When run, the italics option will first ask for the input file. This is the name of the file you wish to italicize. The second prompt is for the name of the output file. The output file can be the same as the input file BUT it is usually necessary to specify a new file due to different size requirements. I'll give you more on that later.

When you use the italic feature, the top dot pattern of each graphic will be shifted to the right by some amount that you will specify (tilt factor). The easiest way to avoid chopping off letters is to do the following:

1. Run the byte dump routine to get the frame size (only the width is needed) of the root file. You could also run option 12, minimum frame, and get the same information.
2. Now, add an obviously large frame width to that number. For example, if the frame width of the root file is 33, double it to 66. This will never pose a problem except in extremely wide fonts where you might hit the 128 max.
3. Use the FORMAT option and create a file that is wider than the root file but uses the same number of printer lines.
4. Next, use the copy routine to copy the root file into the larger file. You will also want to run the MOVE / JUSTIFY routine.
5. When in the move / justify routine, left justify the new large file. I have also found that it saves trouble if you move all letters one dot right after the justification.
6. Now run the ITALIC routine until you get the look you want. When you are satisfied with the tilt of the letters, run option 12 and find out the smallest frame you can get your new font into. Then, create the new file and copy the italic font into it. That is all there is to it and though there are shorter ways, this is the safest.

Now onto the next routine. The following MOVE / JUSTIFY routine is a bit involved so be prepared to sit down at your computer and try to follow along. The routine isn't difficult but there is a lot to it!

CHOICE 7, MOVE / JUSTIFY

I'm going to attack the explanation of this routine in a very systematic manner. Not because it is difficult but rather, because it can do so much. First, I'll list the features. You can CENTER any letter or range of letters within a specified file. You can MOVE any letter any number of dots in any of the four directions within a frame. You can LEFT JUSTIFY the letters (move them all as far to the left as they will go) or you can RIGHT JUSTIFY the letters (move them as far to the right within a frame as possible). Each one of these manipulations is valuable when creating a letterset or graphic. Also, hitting (ENTER) in response to any prompt will skip you to the next prompt in the sequence so you don't have to concern yourself with dealing with an unwanted feature.

Now a look at the program operation. The first prompt is for the INPUT file. As in all cases, that is the source of the letters or graphic you are working on. The second prompt is the OUTPUT file. Again, this can be the same as the INPUT file or it can be a different file with corresponding dimensions.

Next, you will be asked for the LOWEST and HIGHEST letters you wish to work on. This is the same range specification technique we have used throughout these programs. As you can tell, you will be able to work on 1 letter or any range of letters.

Now for the tough part. The next prompt is for LEFT JUSTIFICATION. If you hit the (ENTER) key the feature will be skipped and you will be prompted for the RIGHT JUSTIFY. If you respond YES to the prompt, you will jump onto the next prompt (which is move up / down or left / right). For now let's just consider the JUSTIFY command. If selected, each character in the range will be moved as far to the left as possible. This means that a character such as "! " will be moved farther to the left than a "W". You may see this in your output when you type. For example: this LQ letterset was LEFT JUSTIFIED and as a result, the characters ", ! , and ", were moved as far left as possible. When typed in sequence, the justification makes the ! look closer to the " than to the ". See "! ". This is only a problem for some of the narrower characters. The problem can be corrected by simply specifying the range so that the punctuation characters are not justified.

The RIGHT JUSTIFICATION works in the same way but the justification moves the characters to the right of the frame. The same problems exist in reverse. If you do not choose a JUSTIFICATION feature, you will be asked if you want to CENTER.

The CENTER feature centers the characters from left to right only. There is no up and down centering. If you select LEFT justification, you will not be given a chance to select CENTERING or RIGHT justification. That makes sense doesn't it? Regardless of whether or not you select CENTER, or LEFT / RIGHT justification, or opt for none of these you will get a chance to move UP, DOWN, LEFT, RIGHT.

After getting through the above three choices, and whether or not one of them was selected, you will now be prompted for MOVE (U)p, (D)own or (N)o move. If you select either UP or DOWN, you will be prompted for the number of dots to move the letter. If you select UP, DOWN or NO move, you will still be prompted for the MOVE (R)ight, (L)eft or (N)o move. Again, selecting a positive move will result in a prompt for the number of dots required.

If you wish to EXPAND a letter (a later option) you will need to move the letter exactly 1 dot from the left side of the frame. The easiest way to do this is to LEFT JUSTIFY and then MOVE RIGHT 1 dot. Certain combinations do not work together. For example, the JUSTIFICATION does not function with the MOVE selections. That has to be done in two moves.

You'll get a chance to use this routine in our example later on in the manual. Again, this is very simple but does require a little forethought!

There are only two features of this utility that I dread trying to explain. They are MERGE and WIGGLE. Since MERGE is next I think I'll take a break before I start. I suggest you do the same. Take some time to run through the routines described so far. They are easy to use and will give you a world of power. If you get stuck, look at the example at the end of the manual. It will give you a run through. Also, if you get stuck, I suggest that you go back to GEAP and DOT WRITER and get some practice. These routines are for the EXPERIENCED users and we assume that you know what you are doing. For an experienced user, all of this is child's play!

CHOICE D, THE MERGE ROUTINE

OK! I've put this one off long enough. Here is the MERGE routine. To start with, a brief explanation of what the MERGE is for.

Basically, all the merge command does is ADD or SUBTRACT a specific string from each character in a letterset. If for example, you wanted a permanently underlined font, you could create the underline, save it to a file, then MERGE it into each character of the object letterset using the MERGE routine. Then, each character in the letterset (or and range thereof) would have the underline added to it. In reverse, you could use the MERGE command to remove the underline from the letterset.

How to do it? Well that is what I have been dreading. But, for an example, let's use the PL letterset. By now, all of you know that the PL set is very similar to the normal EPSON (or C. Itoh) letterset. It is based on 10 CPI although the proportional feature may change that number during actual printout. We're going to add a little graphic character (a line with a circle centered on top) to the bottom of each character in that font.

The first step is to find out where to put the graphic string in the frame. That part is easy for you experienced users. Run EXPMOD8 and transfer a lowercase letter (like the "g") onto the screen. This will do two things. First, we get the letterset frame to work in. Second, we find out how low we must go with our graphic to avoid the bottom of the lowest character.

Step two is to create the graphic in the space available in the frame. If you are doing your own graphic, you must think things out. For example, if you want to add a graphic to the side of a character, you can't really use the letter "I" as the frame character. If you do, and you place your graphic too close, the wider letters (like the "W") will be blitzed. Think things out! In our case, we have plenty of room below the lowercase letters to create out little line with a circle.

WARNING! WARNING! WARNING! DON'T TRY THIS ROUTINE ON YOUR ORIGINAL FONT! It can shatter a lot of work if you mess it up. Be sure that you work with a WORKING disk that contains only copies of your good fonts! Now, create the graphic. I'm going to create a line that is the full width of the frame. I am also going to create a little circle on the top of the line and centered in the frame. You can do what you want.

Now that you have created your graphic, it is time to save it. First step! Go into the DESIGNATE mode and then to the "UO" menu. Option #1 allows the operator to enter the file name. The name of the module you want is SVSTRNG. This is an expansion module to GEAP but it is on your MANIPULATION disk. See the list of filenames in the front of the manual if you don't believe me. When you have loaded SVSTRNG (save string) you will be returned to the DESIGNATE mode and your graphic will be returned.

Now you have to find out where the graphic sits on the screen. This is the PRINT AT location and can be obtained by entering the KEYPAD (0) mode and placing the cursor at the portion of the graphic that is farthest left in the frame AND closest to the top of the frame. In other words, the LOWEST screen location occupied by the graphic. Now hit the (ENTER) key and note the PRINT AT location that will appear at the top of the screen. This part may take some practice. I have found that it is easy to erase part of the normal letter by using the wrong PRINT AT location. Fly by the seat of your pants for a while - you'll catch on!

Now make your graphic the DESIGNATED figure. You all know how to do that. BE SURE THAT YOU ONLY DESIGNATE THE NEW GRAPHIC. If there is some overlap, erase the offending parts, including any of the frame boundary. To be on the safe side it is easiest to erase everything except the graphic that you will add (or subtract). Once you have designated the figure you can save it to a file.

This is an easy step. You should have already loaded the SUSTRNG expansion module. If you haven't, do so now. Go to the DESIGNATE mode (you will probably be there by now) and press the "S" key. You will be asked for the name of the OUTPUT file. This is the name that you want to use as a graphic. Since I have created a little LOOP to go with the PL letterset, I'll call my file LOOP. Name yours as you will and specify the disk drive in the name. This is optional but it is a good idea to get into the habit of specifying drive numbers when working with utilities. It keeps you from wrecking a lot of work!

With that done, it is time to run the LETMENU program. Of course you want to select CHOICE #6, MERGE. Once loaded the merge program will ask you for the name of the INPUT (source) FILE. This refers to the letterset that you want to add to or subtract from. In my case it will be PL. The next prompt is for the name of the OUTPUT file. This is the destination name and in my case, since I am adding a LOOP to the PL letterset, I'll use the name LOOPPL.

Now comes the standard part of each of the routines. That is range specification. You are asked for the LOWEST value and the HIGHEST value. As always, you can specify whatever range you want. In my case, I'll specify the entire letterset, "! " is the lowest character and "z" is the highest.

MERGE will also ask for the name of the file that contains the string to be added or subtracted. For the sake of the example, I use LOOP as the string name. So I would answer LOOP: d. Next you will be asked if you want to (A)dd or (S)ubract the graphic string. You might think that DESIGNATING a blank area would be best to remove a string but it isn't necessary. If we add a string, we may want to subtract it too! Just answer "A" or "S". For my font it will be "A" for add.

OK! MERGE is off and running and you see in the upper left of your screen a dazzling display of addition of a character to rapidly flashing characters! BLAHHO! It's done. SAVING appears on the screen and LOOPPL is created. Let's try a test print!

DID IT WORK! YEP, SLICK AS CAN BE!

I can't believe I just got through that. Looking it over - it even makes sense! That is the MERGE routine. With it you can create or remove standard character strings. Now that that is over I only have two more toughies to deal with. I'm not going to jump in though! I want you to spend a little time with this MERGE routine. Think it out. Get the feel. Give yourself some time to understand it and me some time for a drink and some thought before I have to crash into the MAGNIFY / EXPAND routine!

CHOICE 9, MAGNIFY / EXPAND

Here we are again, ready to assault another frontier, climb another mountain, face another adventure and cliche ad infinitum! This little beauty is the MAGNIFY / EXPAND routine. Let's start by explaining what MAGNIFY is in relation to expand!

First, MAGNIFY means to make larger. That is what we do. We make a letter or graphic larger. This routine will increase the size of an entire letterset as fast as Superman can save Lois Lane. The resulting letters may need some cleaning up since magnifying causes a little squaring of the previously round letters. This is much the same as in the SPLIT routine.

EXPAND on the other hand, not only increases the letter size, but also breaks it apart into dots! . Say what? That's right. When we expand we actually pull the dots apart by a specified number of dot widths. This can occur in both directions. You'll get a better idea of this when I EXPAND this LQ font right before your eyes!

But, that is enough for now. I lied before. I said I was going to take a break and I didn't. Now it's 3: 00am and there is a limit to even my dedication. See you in the morning!

Boy! It seems like only a few minutes ago I was saying goodnight and here it is, morning. Well, I have six cups of coffee, a note pad and the kids are fighting - just the right atmosphere to continue our discussion. We are going to start with the MAGNIFY / EXPAND routine aren't we. OK. Let's get to it.

We'll start with MAGNIFY since it is the simplest. General rules first! The MAGNIFY routine ALWAYS gives 2 times magnification. That means that a 10 dot wide by 2 printer line high font will require a frame size of 20 by 4. To create this frame size in a receiving file use option 3, the FORMAT routine. In the case of both the MAGNIFY and EXPAND routines, it is also necessary to move the intended characters one dot width from the full left justified position. So, the first step, if the root font is left justified, is to run option 7 and move the letters one dot to the right. This is only necessary if the letters hit the full left border. It also applies in reverse, if the wide characters hit the right extreme.

In my example I am going to use the LQ font and it is fully left justified. My first stem is to find out the frame size. By now it should be clear that there are several ways to do this. I usually just run EXPMOD8 and transfer a letter to the screen. Then I count the frame size out. The LQ font is 15 dots wide by 2 printer lines (16 dots) high. So, for the magnify routine I will have to make some modifications. The first step is to run the MOVE routine and move every letter in the LQ font one dot to the right.

Next I have to format my files. LQH will accept the magnified version so I format it to 2x15+1 or 31 dots wide by 2X2 or 4 printer lines. A little explanation. While I only doubled the printer lines (2 times magnification) I had to double the width PLUS ADD the 1 dot that I moved the font to the right.

Since I am in the FORMAT routine I might as well format the the EXPAND file as well. I have decided to call the expand file LQW and to expand it 3 times in width and 2 times in height. That means that the file must be 3X15+1 wide - that's 46 dots wide. (Note that I added the one dot for the right move from earlier). And I will make the height 2x2 or 4 printer lines.

Now that I have formatted the files, I can start option 9. First I'll magnify. The first prompt is the INPUT file name -- LQ. The next is the OUTPUT filename - LQH. The range is requested next and we will use the entire set. "! " is the LOWEST symbol and "z" is the highest. OK. We're off! . The expand routine flashes each LQ character, MAGNIFIES it 2 times and then saves it. That's it. The finished product is saved. Let's have a look:

ABC D E F G

You will note that the letters have become a little blocky. Some individua editing may be in order. I'll pass that by for now. It's part of the GEAP program.

Now for the expand. The initial prompts are the same: The INPUT is LQ an the OUTPUT file is LQW. When you select EXPAND you are asked for the number o expansions in the X direction. The X direction represents width. We'll choos three as we planned.

Now we are asked for the expansion in the Y direction. Y represents heigh and is in printer lines. We'll stick with our original plan of 2. We're off again. The EXPAND will print the original character, then show the expand character an finally will save the file. Here is the result of out expansion:

I know, it's nothing exciting here but just imagine what you can do with thi utility and some re-editing. Add a few lines, increase the number of dots in a area for darker patches. Instant art!

That covers the MAGNIFY/EXPAND routine. Next we have to cover the worst o he group - WIGGLE. Wiggle is a bit on the extreme side. It allows you to offse dot patterns in various combinations of sign waves. To be honest, I really don' understand all of its power! To make it easy for me to explain and to give yo something to experiment with, I'll give you a brief explanation of the routine an a few examples. You will have to take it from there and EXPERIMENT. This time i is a cop out. I could load this manual with examples and still not have full explained the function. There are things like UP / DOWN wiggles, LEFT / RIGH wiggles, TAPERED wiggles, MAXIMUM SHIFT, INITIAL SHIFT and WIGGLE SPEED. One thin I will tell you though. The Wiggle routine works best on complicated or larg graphics rather than simple lettersets like the one seen on this manual. For tha reason, I will include a GEAP logo which I will distort in various ways to give yo an idea of what you can do. The rest is up to you. If you come up with particularly interesting result, let me see it and also explain what you did to ge it. I'll stick it into our user library for others to use and as a further guide. O - Here goes!

CHOICE 10, THE WIGGLE ROUTINE

I haven't been looking forward to this discussion but I've always felt that the best way to do work you don't want to do was to jump in and get dirty. Actually, from a user standpoint, the WIGGLE routine is fairly simple. It's explaining it that is tough. I guess that the best way is to start with a normal graphic and then wiggle it a bit to let you see what happens. I'm not even going to try to give you total training. Most will have to be experimentation on your part. Let me see what I can do!

First, the explanation. The WIGGLE routine works best on graphics with full screen (128 x 6) dimensions. That is due to the nature of the routine. Essentially what we do is allow you to add a sign curve to the graphic. The wave effect can run from top to bottom or from left to right. We have also added a taper feature that allows the wiggle to die out before the edge of the screen, thus preventing cut off of the graphic.

You will be prompted for the INPUT (source) and OUTPUT (result) file names as always. The next prompt asks if you want to use the (A)utomatic wiggle or the (R)oll your own option. If you select the (R)oll your own option, you are asked how much you want to move EACH column. That is a big order but you can create some unique distortions of the graphic by using that feature.

If you select the (A)utomatic wiggle you receive more prompting but the answers are easier. First you are asked for how many UP / DOWN wiggles you want. You are also prompted for the number of LEFT / RIGHT wiggles. This allows you to wiggle either way or both ways at the same time.

For each wiggle (UP / DOWN - LEFT / RIGHT) you are asked for the maximum shift. This number is the greatest number of dot offsets you want. I can't think of a better way of phrasing it than that. If you select 10 for example, no dot will be shifted more than 10 dot widths out of place.

The next prompt asks for the wiggle speed. This is the number of peaks and valleys in the curve. The higher the number, the greater the wiggle speed and the more peaks in the finished wiggle.

Finally, you will be asked if you want a tapered wiggle. This is important since without tapering, the wiggle could push the graphic out of the frame and thus, cut it off. Experimentation is the best way to see if a graphic will need taper to look right. The taper feature reduces the wiggle toward the edge of the graphic in order to prevent extending past frame.

That is the wiggle feature. I know it is a bit obscure, but I did warn you about it. I have tried to explain it more clearly, however, the more I explain the more confusing it seems to get. Let me try out some examples on you to see if that will help. The following GEAP logo is done normally. Each letter is approximately 1 screen in size.

Now I'm going to wiggle it. I'll use UP / DOWN only and I'll select 1 as the number of UP / DOWN wiggles. Note that each step in the wiggle is one dot offset. I'll also select a maximum shift of 5. No dot will be more than 5 dots out of place. A wiggle speed of 4 will create 2 peaks and 2 valleys. An initial shift of 0 completes the parameters and the result is:

Now, if I use the same parameters and select a tapered wiggle the result will be:

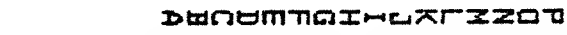
A left / right taper results in the following distortion:

Now let's try a non-tapered UP / DOWN and a tapered LEFT / RIGHT. The interesting result of this combination is:

I'm afraid that's about all I can tell you about the WIGGLE routine. Use it and try different values. It won't take long to catch on and the results are worth it. Also, you can take heart in the fact that the worst of the utilities are over. The remaining ones are straight forward and you will have no more to puzzle over!

CHOICE 11, ROTATE / MIRROR IMAGE ROUTINE

Finally, another simple routine. You will be prompted, as always, for the INPUT and OUTPUT files, and the range of letters to be ROTATED. You will also be asked for the degrees that you want to rotate the object file. Since the TRS-80 (trademark of Tandy Inc.) graphics are not square, the 90 and 180 degree rotation produce the best results. Here is an example of the PL letterset rotated 90 degrees!



Now for the MIRROR letter option. You will be fully prompted for input, and the output will be a mirror image of the object letterset. As always, you should have a file prepared to accept the output from the routine. In the case of mirror image only, the output frame will be the same size as the input frame. In the case of rotation, read on.

There are a few warnings necessary. First, the routine rotates around the center of the frame so you should center the characters in the frame. Menu option 7 will do that for you. Also, The rotation takes room so be sure you have a big enough frame. The easiest way to do that is to create a large frame. Copy the object letterset into it and then center the resulting letterset. When the rotation is done. Use option 7 again and get the characters into the upper left corner of the frame. Finally, run option 12 and find the minimum storage size needed for the new letterset. Use option 3 to format that file and option 2 to copy the rotated letterset into the minimum frame. That's all folks!

CHOICE 12, THE MINIMUM STORAGE ROUTINE

This little beauty will save you a lot of time and it is easy to use. Once selected, this routine asks you for the object file's name. It then asks for the range of characters. In most cases it is best to specify ALL of the letterset so the range would be "! " to "z". The routine then checks every character in the range, selects the largest one and then computes the smallest frame necessary to hold the largest character in the file. The result is printed on the printer.

With routine, it is possible to work in a frame with PLENTY of room and still be able to reduce the frame size to minimum. It is best to have the character set left justified and moved all the way to the top of the frame if you want the smallest possible result.

Once you have the smallest frame parameters, use option 3 to create the file and option 2 to copy the letters into it. Another easy one isn't it?

CHOICE 13, RUN DOT WRITER

This last choice is obvious. Selecting it allows you to enter Dot Writer directly from the manipulation routines. If you need to work in Dot Writer, select this option, do your work in Dot Writer, (BREAK) and then run "LETMENU" to get back to the manipulation routines. This is an easy back and forth procedure for working on your graphics.

Well, that is it for the overview. You have seen how each utility works and how to use them together to manipulate your graphics. You should now get even greater power from your GEAP and Dot Writer than ever before. If you come up with interesting graphics or lettersets, remember we are always interested in ...ng them. Also, if you have ideas for improving the programs please send them ... User feedback supplies us with most of our ideas.

ERROR CHECKING

Just a few words about error checking. When we first gathered together these programs, we didn't worry about error checking. Since they were only used in-house, we didn't think it was worth the time. But, the idea finally came to make these available to our users and so, we had to make them as user friendly as possible. A menu was created to allow for greater selection flexibility; the programs were given a final option to RERUN, RETURN TO MENU or END and some error checking was put in. At the time of this writing, the program has some error checking. The error checking routines are fairly simple and there is no need to document them. If you create an error, you will be flagged and most errors are not fatal (that means most errors will not cause a program crash and you will be able to begin again or return to the menu).

That is it for this manual. I'm sure that you will enjoy the programs and we are anxious to receive input from users telling us what they think about the routines and what types of routines they would like to see in the future!

IN CASE OF PROBLEMS

If you run into problems using these routines, please call Rick McGarvey, 221 Hirschifeld Dr., Williamsville, NY 14221 (716) 634-3026. I try to keep normal business hours but my schedule prohibits me from being on hand all the time. If you get the answering machine, don't be shy. Leave your name and address and your phone number and a description of your problem. You'll have plenty of time so there is no need to rush. In turn, I'll get back to you as soon as possible with information on how to defeat the difficulty you are having. Thanks!

INDEX

RCM COMPUTERS
221 HIRSHFIELD DR.
WILLIAMSVILLE, NY 14221
716-634-3026

Patches

Many of our users want patches to fill special needs. In response, we either create the patches ourselves or supply patches that have been submitted by users. The following patches are offered for your use. Understand that though we have created and tested all of the patches offered here, we offer them for use at your risk and cannot warrant them in any way.

MAX 80 Patches

The MAX80 "looks" like one computer but prints like another. That throws the program. In the new version, the patch is simple. Add one line to WP/ITO or WP/EPS respectively:

WP/ITO Add:

```
1 POKE &HCD8B,0: POKE &HCD8C,0: POKE &HD452,0: POKE
&HD453,0: POKE &HD4D3,0: POKE &HD4D4,0
```

This line, when added to the beginning of WP/ITO will make the MAX80 operate correctly with Dot Writer.

WP/EPS Add:

```
1 POKE &HCAA0,0: POKE &HCAA1,0: POKE &HD137,0: POKE
&HD138,0: POKE &HD1B8,0: POKE &HD1B9,0
```

This line, when added to the beginning of WP/EPS will make the MAX80 operate correctly with Dot Writer.

Letterset Data Sheet

(NR) = Not Recommended and YY+x = letter Height + descender

Font #	Font Name	Frame Height dots,lines	Vert. Letter Size	Suggested .UP Value
	FF	16,2	9	16
	MB	16,2	11+4	1(NR)
	MB2	32,4	22	1
	BB	24,3	16+6	1
	BO	24,3	16+6	1
	SE	16,2	9	32
	MP	8,1	5+2	1(NR)
	OE	16,2	16	1(NR)
	PL	16,2	7+2	32
	MC	16,2	15	1(NR)
	GR	16,2	9	32
#1	SHADED	16,2	13	2
	CS	16,2	10+5	1(NR)
	SHADOW	16,2	16	1(NR)
	CM	16,2	12+4	1(NR)
	MB3	32,4	24	64
	FANCY	24,3	24	1(NR)
	BALL	24,3	24	1(NR)
	BW	24,3	24	1(NR)
	SF	16,2	9	32
	STENCIL	16,2	11	8
#2	ISHADED	16,2	13	2
	ILETQUAL	16,2	10+3	2
	IBO	24,3	16+6	1
	IMB	16,2	11+4	1(NR)
	IPL	16,2	7+2	32
	IFANCY	24,3	24	1(NR)
	IBALL	24,3	24	1(NR)
	ISE	16,2	9	32
	ISF	16,2	9	32
	IFF	16,2	9	32
#3	SCOMP	32,4	19+8	8
	ANT	32,4	26	16
	FSC	24,3	12+8	4
	HWT	24,3	12+9	2
	LMES	32,4	18+6	64
	LQ	16,2	10+4	1
	ILQ	16,2	10+4	1
	MES	16,2	9+3	4
	LED	8,1	7	1(NR)
	SCS	8,1	7	1(NR)
#4	SGOTHIC	32,4	31	1(NR)
	GBB	32,4	31	1(NR)
	GOTHIC	16,2	16	1(NR)
	LOGO	16,2	14	1
	PB	24,3	18+5	1(NR)
	BB	24,3	18+6	1(NR)
	SMES	24,3	21	2
	COM320	24,3	13+6	8
#5	HUGE	48,6	47	1(NR)
	SHEER	48,6	47	1(NR)
	HLINE	48,6	47	1(NR)
	LCD	16,2	14	1
	FPL	16,2	7+2	32
	CLAR	8,1	7+1	1(NR)
	ICLAR	8,1	7+1	1(NR)

RCM COMPUTERS
221 HIRSHFIELD DR
WILLIAMSVILLE, NY 14221
716-634-3026

NewScript Filename Transfer

Many of our users appreciate the power and ease of the NewScript editor but miss the ability to transfer the current filename from NewScript to DOTPRINT. With the following patch, the transfer will be made in exactly the same manner as with NewScript. The patch shown is for the C.Itoh version of DOTPRINT, but the only difference is that in the EPSON version, the first line to be changed is # 125, not 20 as shown below. Everything else is the same.

To the WP/ITO program, alter line 20 to read as follows:

```
20 GOSUB13000:Z1$=A$:PRINT:IFSF$=""THENLINEINPUT"ENTER I. D. OF FILE TO BE PRIN
ED (0=CANCEL):     ";A$:GOSUB12000:IFA$=""THENA$=Z1$:OPEN"I",1,A$:LINEINPUT#1,EX
:PRINTEX$ELSEOPEN"I",1,SF$:LINEINPUT#1,EX$:PRINTEX$
```

In the WP/EPS program make the same change but the line number is 125!

Now, in both WP/ITO and WP/EPS, add the following subroutine:

```
13000 ' NewScript File Xfer
13010 F=PEEK(&H4016)+256*PEEK(&H4017)-(256*256)
13020 F$="":FORI=18TO41:J=PEEK(F+I):IFJ<>32THENF$=F$+CHR$(J):NEXTELSEI=41:NEX
13030 IFF$=""THENRETURNELSEPRINT:PRINT"Current File is "F$
13100 IFA$=""THENA$=F$:RETURN
13110 RETURN
```

Command	Default	Action
.AJ		Adjust left margin
.AL 'filename'		Alternate Font select
.AP 'filename'		APend file
.BF 'filename'	BF1	Begin Font (BF1 standard)
.BM n	6	Bottom Margin
.BR		Control BReak
.BT 'title'		Bottom Title
.CB n,w or -n,w		Column Begin
.CC n		Conditional Column
.CE on,off	off	CEntering
.CM		Comment
.CO on,off	on	COncatenation
.CP n	2	Conditional Paragraph
.CW on,off	off	Center Width
.DA on, off	off	DArk dot fonts
.DA 0,1,2,3	0	DArk printer fonts
.DS on,off	off	Double Space
.EM on,off	off	EMphasized dot print
.EN		END text file
.ES 'symbol'	!	EScape code
.FO on,off	on	FOrmat
.FM n	1	Footing Margin
.HL		Horizontal Line
.HM n	1	Heading Margin
.HY symbol on,off	shift 3 on	Conditional HYphen
.IG on,off,1,2	off	IGnore text
.IM 'filename'		IMbed file
.IN n	0	INdent
.IX 'text'		INdex entry
.JU off,right	on	JUstify text
.KE		KEyboard entry
.LH n	12	Line Height
.LL n	70	Line Length
.LS n	0	Logo Space
.MF n	1	Magnify Factor
.MX 100,80	80	printer width EPSON
.MX-80		Force 1/72 feed '
.OF n	0	OFfset
.PA		PAge force
.PI 0,1,2,3	2	PItch C.Itoh
.PL n	60	Page Length
.PN on,off	on	Page Number
.PP n	5	Paragraph
.PR on,off	off	Proportional
.PS symbol	#	Page number Symbol
.RD n		Form letter ReaD comm
.RD n (or c) 'filename'		
.RE 0,1,2	0	REverse print
.SD n	0	Space Dots between le
.SK n, -n	0	SKip line, reverse sk
.ST 'message'		STop
.SW n	.5	Space Width
.TC 'phrase'		Table of Contents ent
.TM n	6	Top Margin
.TR a,b,c	0	TRanslate command
.TT 'title'		Top Title
.TF 'name'		Title Font
.UP n	1	Underline Print
.VL on,off,1,2,3		Vertical Line
.VT, .VT n		Vertcial Tab # of lin
.WP		Wide Paper C.Itoh

ASCII ORDER

VAL / CHAR	VAL / CHAR	VAL / CHAR
32 SPACE	63 ?	94 ^
33 !	64 @	95 _
34 "	65 A	96 `
35 #	66 B	97 a
36 $	67 C	98 b
37 %	68 D	99 c
38 &	69 E	100 d
39 '	70 F	101 e
40 (	71 G	102 f
41)	72 H	103 g
42 *	73 I	104 h
43 +	74 J	105 i
44 ,	75 K	106 j
45 -	76 L	107 k
46 .	77 M	108 l
47 /	78 N	109 m
48 0	79 O	110 n
49 1	80 P	111 o
50 2	81 Q	112 p
51 3	82 R	113 q
52 4	83 S	114 r
53 5	84 T	115 s
54 6	85 U	116 t
55 7	86 V	117 u
56 8	87 W	118 v
57 9	88 X	119 w
58 :	89 Y	120 x
59 ;	90 Z	121 y
60 <	91 [	122 z
61 =		

SUGGESTED .UP VALUE FOR UNDERLINING FONTS

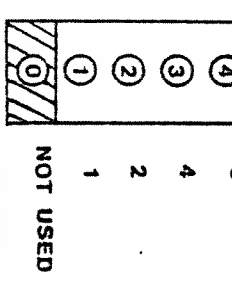
FILE / .UP VALUE

OE / 1	BO / 4
BE / 32	FF / 8
PL / 64	MP / 2
MB / 4	MPI / 2
MB2 / 8	MC / 1
BB / 4	GR / 64

CONTROL CODES

!/ . . . . Begin alternate font.
!7 . . . . End alternate font.
!8 . . . . Underline non-blank
!3 . . . . Underline blank & non-blank
!4 . . . . End underlining.
!(. . . . Begin double width mode.
!) . . . . End double width mode.

PRINT HEAD PIN LAYOUT



Tiny GEAP Commands

S....Save character to disk
T....Load character from disk
W....Write to disk
A....Set frame dimensions
" " ....Go to flashing dot cursor
F....Draw ro redraw the frame
H....Display storage limits
L....Display commands
E....Exit
ARROWS..Move Flashing minus

When DOT CURSOR is on

ARROWS..Move cursor - erase
(SHIFT) & ARROWS..Move cursor - draw
<CLEAR>..Clear character
-...Return to minus cursor
0-7..Adjust cursor speed